AN INTRODUCTION TO DATA BASE MANAGEMENT

Joan C. Veim

CENTER FOR RESEARCH ON INFORMATION SYSTEMS

Computer Applications and Information Systems Area
Graduate School of Business Administration
New York University

Working Paper Series

CRIS #9

GBA #80-109(CR)

Ms. Veim's current address:
New York University, Graduate School of Business

Permanent address:
Institute for Computer and Information Sciences
University of Bergen, 5014 Bergen-U, Norway

This material has been prepared for presentation in Beijing, People's Republic of China, August 1980. The paper provides an overview of the principles and theories underlying the development of current data base management systems. August 1980.

The Evolution of Data Base Management Systems

Data base management systems have evolved from the file management systems developed following the addition of direct access storage devices to computer systems in the mid 1960's. These devices allowed the selection of a single record from any location within the storage device given its unique storage address. The main objective of both file and data base management systems is to provide generalized data access and manipulation for the data associated with an application system.

File management systems provide data access to the records of one or more files from an application program. The files supported by a file management system may have differing structures, i.e. the physical organization of the record occurrences of the record-type constituting the file's data. A file structure aims at optimizing data access to one or a sub-set of the records of the file. The more complex of these structures supplement the user data with control data in the form of indicies or pointer sets to aid the data access.

The basic file structures are illustrated in figure 1 and presented in some detail in [Veim, 1980]. These structures include:

1. Sequential files

   in which the physical sequence of the record occurrences corresponds to the ordered sequence of the data values of one of the attributes of the records of the file.

2. Direct files

in which record placement is calculated by a user provided conversion routine which calculates a storage address from a data value, normally a unique identifier such as the American social security number.

3. Index Sequential files

in which the user data is stored sequentially according to the data values of the record identifiers or keys. The user data is logically divided into sub-files and an index is generated with the high key value from each sub-file and the address of the sub-file. Location of a single record within the indexed file requires a search of the index followed by a search of the relevant sub-file.

4. Linked files

in which the data records within the file are connected by pointers which form record chains joining those records belonging to some logical sub-set of the data. There may be many separate chains of records within the file and any one record may be a member of several of these chains. In linked files the pointers are stored within the record area of the separate records.

5. Inverted files

in which data records belonging to a sub-set of the data, particularly containing a common data value, are identified by

pointers stored in a file directory with the data value identifing the record set.

The file structures each support a specific access type:

sequential access:	sequential and index sequential files
single record access:	direct and index sequential files
group record access:	linked and inverted files.

The choice of file structure for a specific data set depends on the access requirements of the application system.

Development of the software to provide file management was divided between the computer vendors, who have traditionally supplied sequential, direct and index sequential file management software and the users who were left to develop systems to support group access. This has resulted in a proliferation of data management systems, the most advanced of which received the name of data base management systems as early as 1967 [Bleier, 1967] with TDMS a Time-shared Data Management System.

The availability of generalized data management systems allowed application systems to increase the number of files which could be processed. However the diversity of the software lead to the development of independent application systems using, in many cases, overlapping data, as each system tended to use its 'own' data management routines and more importantly its own definition of the file structures. This situation lead to extensive data redundancy among the application

systems of the same organization.

DATA MANAGEMENT REQUIREMENTS:

In the late 1960's, so many users/owners of multiple application systems each with multiple files, were experiencing problems with data management that several committees were formed to review the state-of-the-art of data management software [CODASYL, 1969], develop data management requirements [Guide-Share, 1970] and suggest a standard for data base management system structures [CODASYL DBTG, 1971]. From the reports of these committees, the development work of the major computer vendors and proposals from the user and research communities a consensus of what data base management systems should be has evolved.

Briefly a data base management system, DBMS, should provide data management facilities for inter-related data types for multiple data users. The major objectives of a DBMS are to provide:

1. data storage and retrieval
   for inter-related data of multiple record types.
2. definition and support for inter-data relationships,
3. support for multiple access paths from multiple users,
4. reduction of data redundancy,
5. support of multiple users while providing data privacy and integrity.

DBMS Architecture

Data base management systems differ from file management systems in two important areas: the data types administered and the importance placed on the data resource. A DBMS is designed to administer data representing two or more inter-related objects as two or more record types with DBMS maintained relationship data, while the file management system establishes separate files for each object/record type and leaves the generation and maintenance of the inter-relationships to the user application programs.

Secondly, in DBMS environments, the data is assumed to be the central area of concern, while the current application programs are assumed to be a sub-set of the possible users of the data. Thus the DBMS tends to structure the data for future as well as current usage. In file management system environments, the application program set is assumed to be of central concern, while the file structures selected are intended to optimize processing of the current applications. Future users must conform to the current structures.

The shift of emphasis from program management to data management is illustrated in figure 2. One consequence of this shift is that the DBMS must maintain a more detailed description of the data, its inter-relationships, usages and users than the file management system. This enables the DBMS to support more complex data retrieval as well as provide a higher level of data usage monitoring and security control for the shared data within the data base. Another consequence is that the

DBMS environment facilitates the addition of new applications and modification of existing ones.

DBMS PLACEMENT:

A DBMS is a service system, which exists to facilitate data management. As such it can be considered a part of the operating system software and is frequently offered as a software product in line with the system's compilers. The user application programs may or may not make use of the DBMS. Figure 3 illustrates the placement of a DBMS within the computer system software. Generally, a DBMS will make use of the existing system's file management routines, particularly the direct access file structures.

DBMS - USER INTERFACES:

The user interface offered by a DBMS ranges from supplementary programming statements to natural language. The interface language provided is determined by the system constructor's view as to who is to use the DBMS. DBMSs have been classified by their interface languages into:

1. Host language systems,

   in which user access is through DBMS access statements imbedded in a programming language, most commonly in COBOL and PL/1 though FORTRAN and assembly language interfces are also available.

Popular systems of this type are IBM's IMS [IBM, 1971], UNIVAC's DMS-1100 [UNIVAC 1978], MRI System 2000 [MRI,1972], IDMS [Cullinane, 1975] and TOTAL [Datapro 1972].

2. Self contained systems,

   in which the user specifies his/her data request through a DBMS language without using a programming language. The interface language is a fairly restrictive subset of English (only because these systems have been constructed for the English speeking market).

   Example systems include: ADABAS [Software AG, 1971], SEQUEL [Chamberlin and Boyce, 1974].

This classification of DBMS types has only limited validity as the differentiation stems from the particular system's view as to who is to be the primary user of the DBMS, an application programmer or the 'end user', (the user of the data itself). As experience with DBMSs increases many systems have supplemented their user interface to service multiple user types.

DBMS STRUCTURE:

A DBMS can be considered as a system with three basic components, as illustrated in figure 4. Not all DBMSs are clearly divided in this fashion, however the functions implied must exist. These basic components are:

1. Data Base Management Routines, DBMR,

   which include all the software required to provide data management for a data base. Primary facilities of the DBMR are:

   1. a data definition language, DDL,

      which is used to define the data elements, record-types and inter-record relationships which are to be stored within the data base,

   2. a data manipulation language, DML,

      which provides the language interface between the user and the DBMS,

   3. data storage and retrieval,

   4. access control,

   5. back-up and error recovery and

   6. data base structure maintenance.

2. a SCHEMA *)

which contains the data and user definitions required by the DBMR to service the user requests via the DML statement structures. Generally the SCHEMA must provide information about the data base content at three separate levels:

1. the conceptual schema

   describes the total, logical data content of the data base, without details of the current physical layout of the data,

2. the external schemas

   describe those data required for each user or application. There will be as many external schemas as there are diverging usages of the data base data. Each external schema must be consistent with the conceptual schema.

3. the internal schema

   describes the current layout of the data base, including a description of the indices and relationship data which exist.

The schema data can be fairly simplistic, corresponding to the data definition section of a COBOL program or can be quite complex, encompasing the function of a data dictionary [Veim,1979/sec.4 'DB Schema Components'].

---

*) The term SCHEMA as a DBMS component originated in the CODASYL DBTG proposal [CODASYL DBTG, 1971]. There has been much debate on the content and structure of the SCHEMA. The SCHEMA sections here correspond to the ANSI proposal [ANSI/X3/SPARC, 1975].

3. the data base,

which contains the user data as described in the SCHEMA as well as any indices and relationship data generated by the DBMR structuring routines. The structure of the data base is determined by the SCHEMA descriptions and the structuring algorithms of the particular DBMS. The data base structure is generally built on the concepts of linked and inverted files.

1. Record occurrences

of each particular record-type are either stored consecutively via direct or sequential structures or linked together forming a linked structure.

2. Inter-record type relationships

are represented by link structures, less frequently by consecutive placement or inverted indices.

3. indices

for access to single record occurrences, are most frequently implemented using inverted structures.

A DBMS requires a data base administrator, DBA, whose primary function is to define the data base content and structure and provide general administrative services to the user community. The DBMR/DDL provides a tool for the DBA for the definition and maintenance of the system SCHEMA. The actual generation and maintenance of the data base structures is a function of the DBMR storage routines.

DB MODELS:

The conceptual SCHEMA provides a total view of the data base content and the union of logical relationships desired by the user community. This schema level is actually a descriptive model of the data within the data base. Prior to defining the conceptual schema, via the DDL, a more general data model for the proposed data base is constructed. A number of structures for this DB model have been proposed of which three have made significant impact on the design of DBMS software. These models, illustrated in figure 5, are:

1. Hierarchic models

   implemented in IBM's IMS and System 2000 (among many others), which require that the data be ordered by record/object-type in a strict tree structure. Access to any record within the data model is through the parent hierarchy from the root level.

2. Network models

   proposed by CODASYL's DBTG and implemented by UNIVAC's DMS-1100 and TOTAL (among many others), allow the data, ordered into record-types, to be inter-related in a network structure, i.e. allowing the definition of multiple access paths to any record.

3. Relational models

   proposed by Codd [Codd,1970] and implemented in SEQUEL [Boyce and Chamberlin 1973] as well as used in ADABAS, require the definition of a base set of relations describing the

fundamental interdependencies of the data elements or attributes. Thereafter any valid combination of attributes may be specified for retrieval and manipulation. The set of base relations form the DB model for the data base.

The objective of the data model is to record, as completely as possible, the logical content and relationships of the data for the data base. Secondly, the model provides a control that the relationships defined are consistent. DBMS implementation tends to follow the implementor's idea of the 'best' data model and the resulting data base will reflect the complexities and costs imposed by the model types.

A hierarchic data base is the most straight forward for implementation, as the primary relationships can be implemented using consecutive storage thus reducing the requirement for supplementary links. However the resulting data base is rigid and difficult to modify. Searches for data along secondary structures are not supported.

The network data base requires relatively extensive use of linkages, a link path for each relationship. The indices supporting access to individual records use inverted structures. Thus the generation and maintenance of the network data base is more complex than the hierarchic, however multiple access requirements are supported.

The relational data base is, conceptually, the least complicated, as each relation (record-type) is treated as a file and the record occurrences stored consecutivley. Access is in principle unrestricted, however the selection algorithms frequently require full data base searches. In practice a relational data base will employ both links and indices to improve access performance and reduce data redundancy. The resulting structure maintenance is comparable to that required for network structures.

Current Trends

None of the existing DBMSs fulfill all of the desired requirements for data management. Most particularly the user interface needs improvment. Also required are aids for determination of good data base structures for a given application. Each of the three major DB model types, with their supporting DBMSs offer good facilities and we will see a trend towards incorporating these good features in future systems. Particularly work is ongoing with:

1. DBMS support of multiple DB models

   Particularly within the user community for a particular data base, there is a requirement to be able to define the separate external schemas according to a DB model which 'fits' the particular application or functional area.

2. multiple user interface languages

Again the data base user environment ranges from the DBA, through application programmers to the end users who process the data via enactment of transactions, statistical analysis and general information retrieval or report preparation. Each of these functional areas has different requirements for a preferable data language.

3. increased SCHEMA sophistication

Particularly incorporation of data dictionary facilities with accompanying query language.

4. DBMS control of data structure generation and maintenance

Which today is a manual task of the DBA but which can be effectively monitored by the DBMR. Also DBMS support for extending the data coverage of the data base, i.e. support for the addition of new data types and relationships.

5. DB design aids

Linking DB design to general system's analysis.

6. Data base machines

or 'back-end' machines are being developed to off load the central computer for general data administration, particularly lenghty logical searches through a data base. The DB machine is functionally similar to the 'front-end' machine handling terminal and peripheral units for a central machine.

A DBMS is a data management facility which has evolved as computer facilities have become more readily available, both in cost and data storage capabilities. Their further development is dependent on increased data processing sophistication of the end users.

BIBLIOGRAPHY and REFERENCES

ANSI/X3/SPARC, 1975
Study Group on data base management systems: Interim Report, FDT 7:2, ACM, New York

Bachman, C.W., 1969
Data Structure Diagrams, Data Base, vol.1, no.2

Bleier, R.E., 1967
Treating Hierarchical Data Structures in the SDC Time-Shared Data Management System (TDMS), Proc. ACM National Conferance.

Boyce, R.F. and Chamberlin, D.D., 1973
Using a Structured English Query Language as a Data Definition Facility, Tech. report RJ1318, IBM Research, California

Chamberlin, D.D. and Boyce, R.F., 1974
SEQUEL: A Structured English Query Language, Proc. ACM SIGMOD Workshop on Data Description, Access and Control, ACM, New York

Champine, G.A., 1978
Computer Technology Impact on Management, North-Holland, Amsterdam

CODASYL, 1969
A Survey of Generalized Data Base Management Systems, CODASYL Sys. committee tech.rep., ACM, New York

CODASYL DBTG, 1971
CODASYL Data Base Task Group Report, Conference Data System Languages, ACM, N.Y.

Codd, E.F., 1970
A Relational Model of Data for Large Shared Data Banks, CACM, vol.13.

Culline Corp., 1975
Integrated Database Management System (IDMS) system publications

Datapro Research Corp., 1972
TOTAL, Cincom Systems, Inc. Datapro 70

Date, C.J., 1977
An Introduction to Database Systems, 2nd Edition, Addison-Wesley, Mass.

Guide-Share, 1970
Data Base Management System Requirements, Joint Guide-Share Data BAse Requirements Group, New York

IBM, 1971
Information Management System IMS/360, Application Description Manual (Version 2), GH20-0765-1, IBM Corp., White Plains, N.Y.

Kroeenke, D., 1977
Database Processing,
SRA Inc., California

Lefkovitz, D., 1969
File Structures for On-Line Systems,
Hayden Book Co. Inc., New Jersey

Martin,J., 1973
Security, Accuracy and Privacy in Computer Systems,
Prentice-Hall, N.J.

Martin,J., 1977
Computer Data-Base Organization, 2nd edition,
Prentice-Hall, N.J.

MRI Systems Corp., 1972
System 2000 publications
MRI Systems Corp., Texas

Palmer, I., 1973
Database Management, SCICON, London

Software AG, 1971
ADBAS publications
Software AG, West Germany

Sundgren, B., 1973
An Infological Approach to Data Bases,
National Bureau of Statistics, Sweden.

Tsichritzis, D. and Lochovsky, F., 1977
Data Base Management Systems,
Academic Press, New York

Ullman, J.D., 1980
Principles of Database Systems,
Computer Science Press, Maryland

UNIVAC, 1978
DMS-1100 manuals
Sperry Univac Corp., Michigan

Veim, J.C., 1977
On Data Base Theory,
working paper, Institute for Information Science,
University of Bergen, Bergen, Norway

Veim, J.C., 1979
DDBS a Dynamic Data Base System - selected papers,
Institute for Information Science,
University of Bergen, Bergen, Norway

Veim, J.C., 1980
An Introduction to Information and File Structuring,
Universitetesforlaget, Norway

FIGURES

Rec.no.	<s-id, name, course, other data ...>
1	1234 : ADAM : SAD :
2	5678 : BRUCE : DB :
3	9012 : CLARK : SAD :
4	3456 : DAVID : DB :
5	7890 : ERIK : PLC :
	. . .

Figure 1a: Sequential file (on name)

Rec.no.	<s-id, name, course, other data ...>
1	7890 : ERIK : PLC :
2	
3	9012 : CLARK : SAD :
4	
5	1234 : ADAM : SAD :
6	
7	3456 : DAVID : DB :
8	
9	5678 : BRUCE : DB :
10	

Figure 1b: Direct file *)

---

*) The transformation algorithm used for this file takes the last digit of the key field and adds 1, using the result as the record address. Note that with this algorithm, all keys ending with the same digit will be synonyms, i.e. be assigned to the same address.

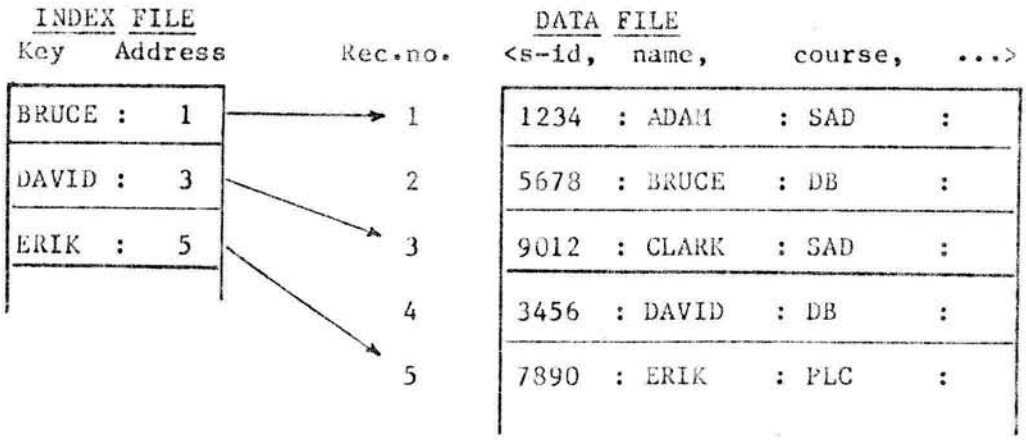


Figure 1c: Index Sequential File (on name) *1)

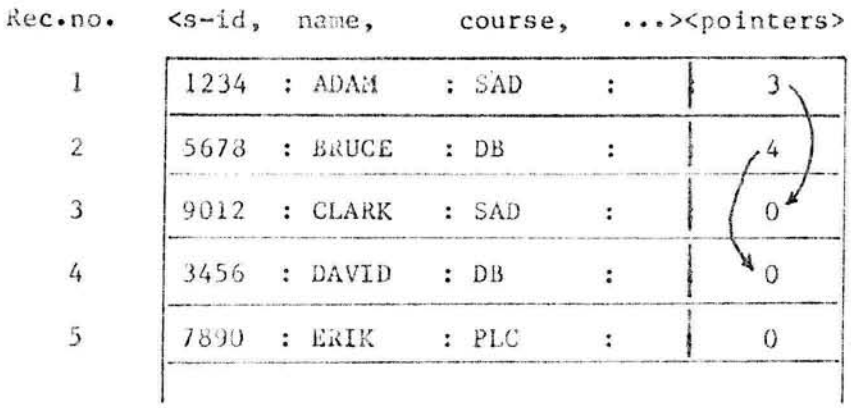


Figure 1d: Linked File (on course) *2)

---

*1) The block size for this file contains two records. The overflow area for new records is not shown, but would be a separate area, normally with chained records and an overflow address in the index.

*2) There are three 'course' chains for the three courses represented in the file; SAD, DB and PLC. The pointer in the last record of the chain is set to 0.

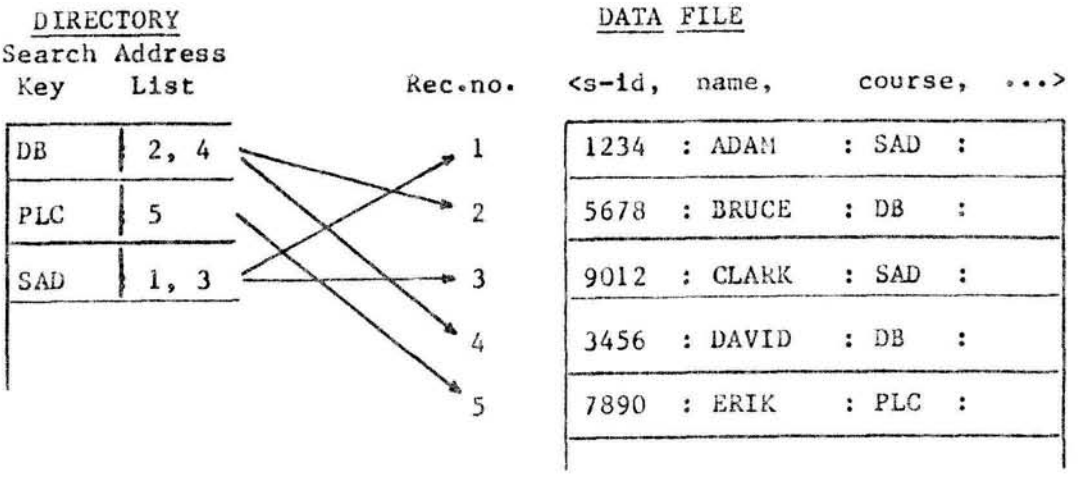


Figure 1e: Inverted File (on course)

FIGURE 1: FILE STRUCTURE EXAMPLES

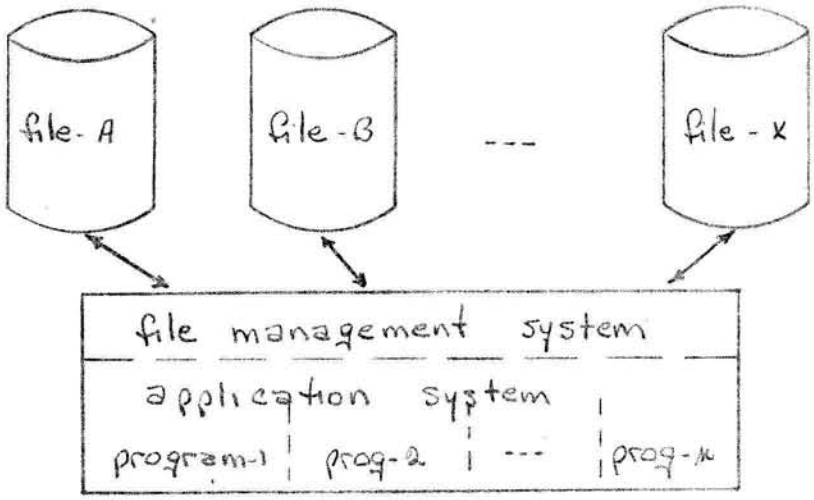


Figure 2.a: Application Oriented System Structure

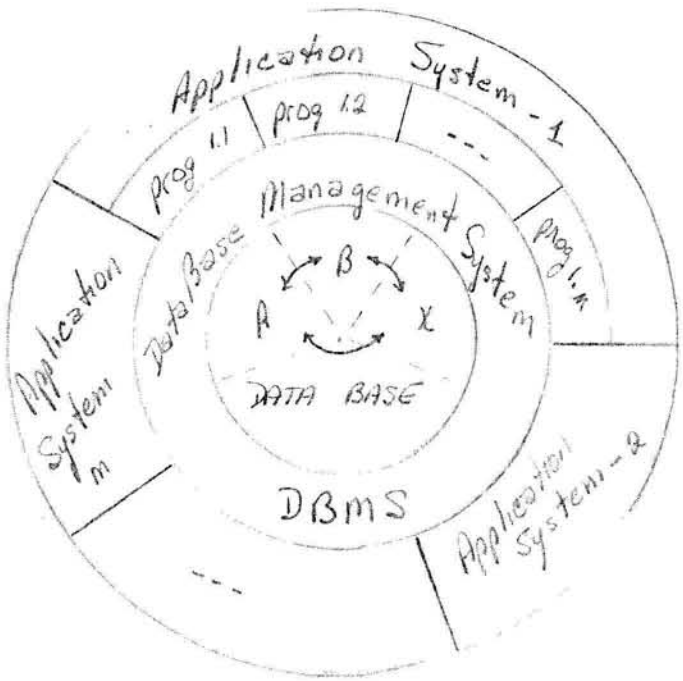


Figure 2.b: Data Oriented System Structure

FIGURE 2: DATA MANAGEMENT SYSTEM STRUCTURES

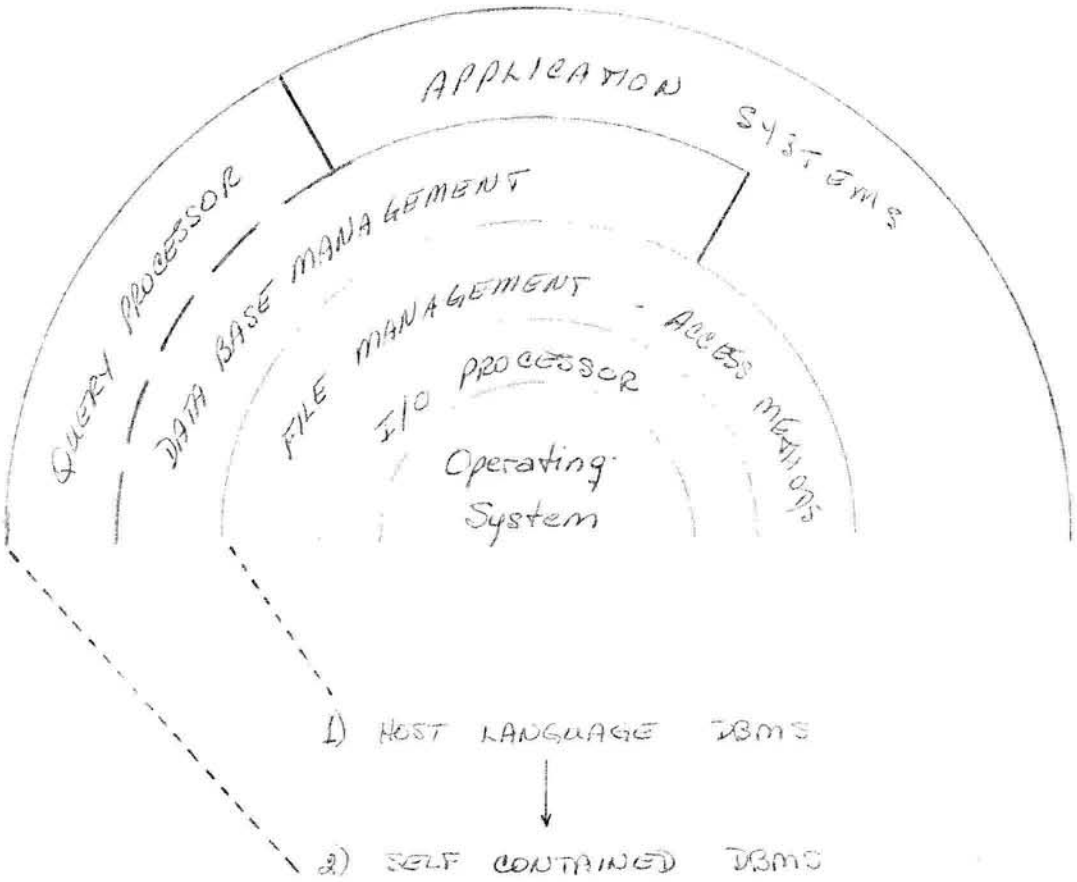


FIGURE 3: DATA BASE MANAGEMENT - SERVICE SYSTEMS HIERARCHY

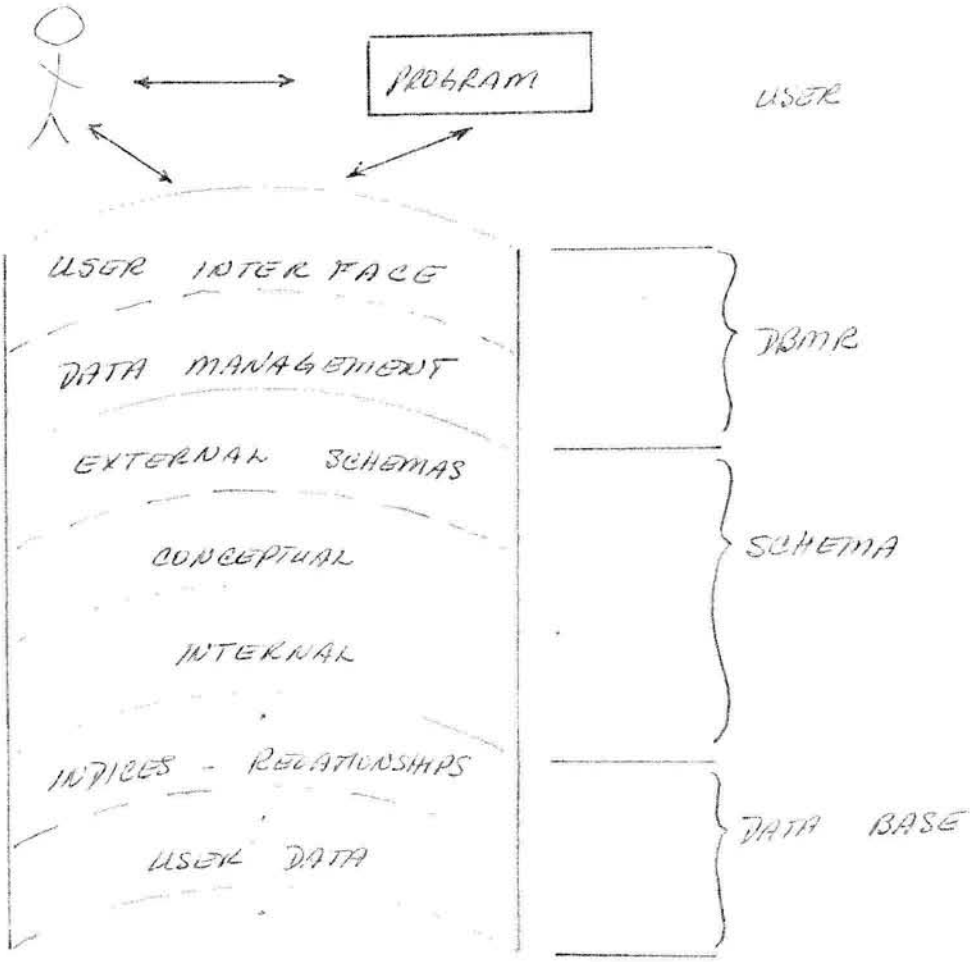
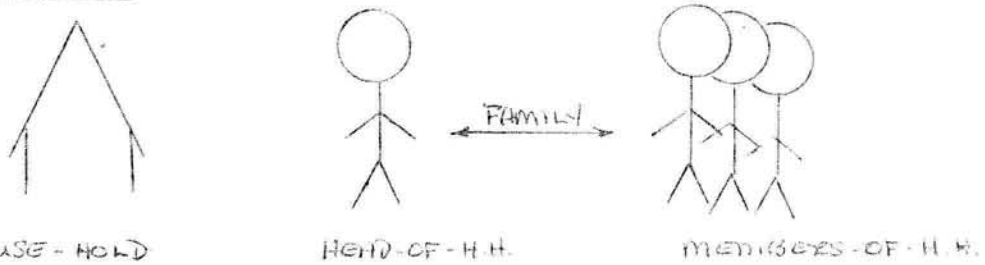
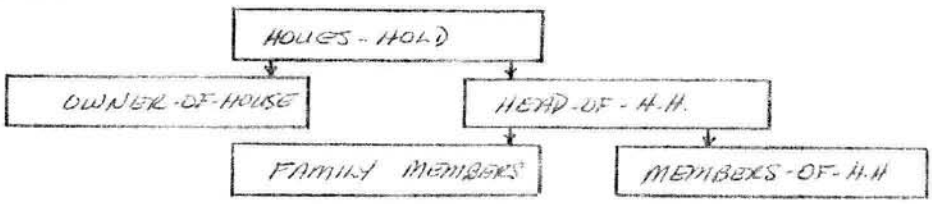


FIGURE 4: DATA BASE MANAGEMENT SYSTEM STRUCTURE

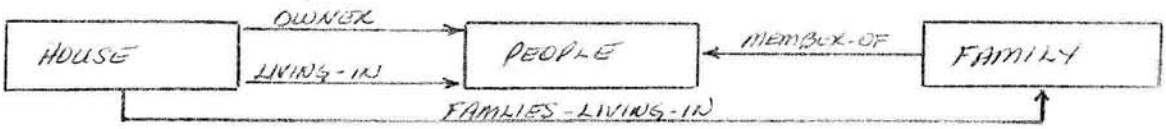
OBJECT SYSTEM



HIERARCHIC MODEL



NETWORK MODEL



RELATIONAL MODEL

PERSON < P-ID, NAME, ADDRESS, H-H-ID, ---- >

HOUSE < ADDRESS, TYPE, #HOUSE-HOLDS, ---- >

HOUSE-HOLD < H-H-ID, NAME, ADDRESS, #MEMBERS, ---- >

LIVES-IN < P-ID, ADDRESS, DATE >

FIGURE 5: DATA MODELING